Benjamin Bytheway

ISYS 540

Dr. Gove Allen

April 12, 2011

Final Project: MSDNAA Downloader

Executive Summary

Background

Every student that goes takes any class from the Information Systems (ISYS) department or the Computer Science (CS) department receives a username and password to the Microsoft Developer Network Academic Alliance (MSDNAA) site.

The MSDNAA is a program that Microsoft created for academic organizations (mainly colleges and universities) to acquire licensed copies of the popular Microsoft software. The software is normally obtained through a service run by e-academy or through a dedicated website run by the school or organization.

Currently BYU subscribes to this MSDNAA program. Because of this, every student has the ability to download and obtain licenses for the many incredible software packages available. They can be accesses by going to the download site for BYU which is located at https://msdn05.e-academy.com/elms/Security/Login.aspx?campus=byu_accinfo.

Problem

The problem comes in the difficulty in downloading the software. It is easy to login and to download individual copies of the software that the site provides, but it is difficult to manage all of your downloads. Also, because the individual subscription lasts only until you graduate, it is important for you to download everything before you graduate. That becomes a difficult thing to do when there are over 200 products to download.

In summary the problem can be summarized as follows:

1. Difficulty in managing downloads
2. Tedious process to download products
3. Long process to download products
4. Subscription period is too short

Solution

With these problems, the downloading process needed to be easier and better managed. So, I came up with a solution that would include one of Microsoft's products – Excel.

The solution was to automate the process of downloading products from the MSDNAA website. That included the process of:

1. Logging in
2. Recording previously downloaded items
3. Adding the products to the online cart
4. Checking out
5. Recording the information needed for each product:
    a. The product name and version (and a picture to associate the product with)
    b. The license key if it has one
    c. The download location

All of this is done in one location to remove the difficulty in finding product keys, etc.

This is done by simply opening an Excel spreadsheet and entering your information like username and password. Once that information is filled out on the user form, the Excel program does the rest.

This solution saves the student around 100 hours of time so that they don't have to waste their time clicking through things that they would already accept anyway.

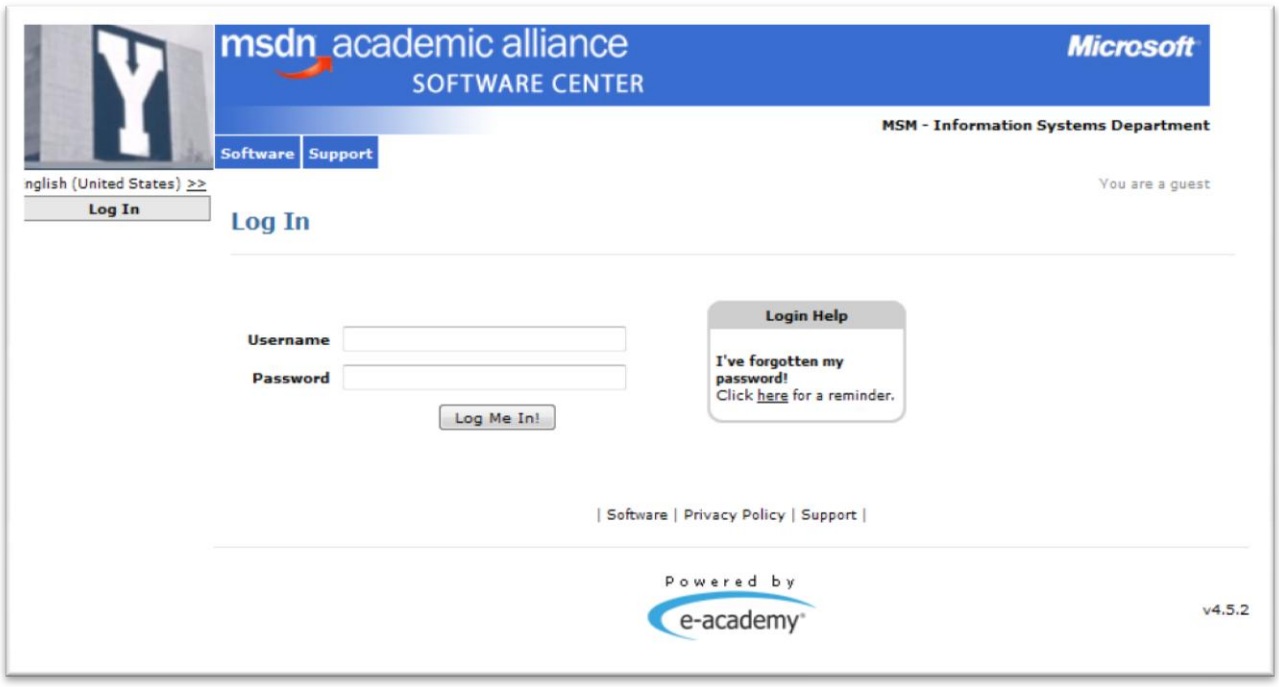
Implementation documentation

In the following section, I will describe what I did to implement the solution. You may notice that the outline will follow the process outlined in the "Solution" section of the executive summary.

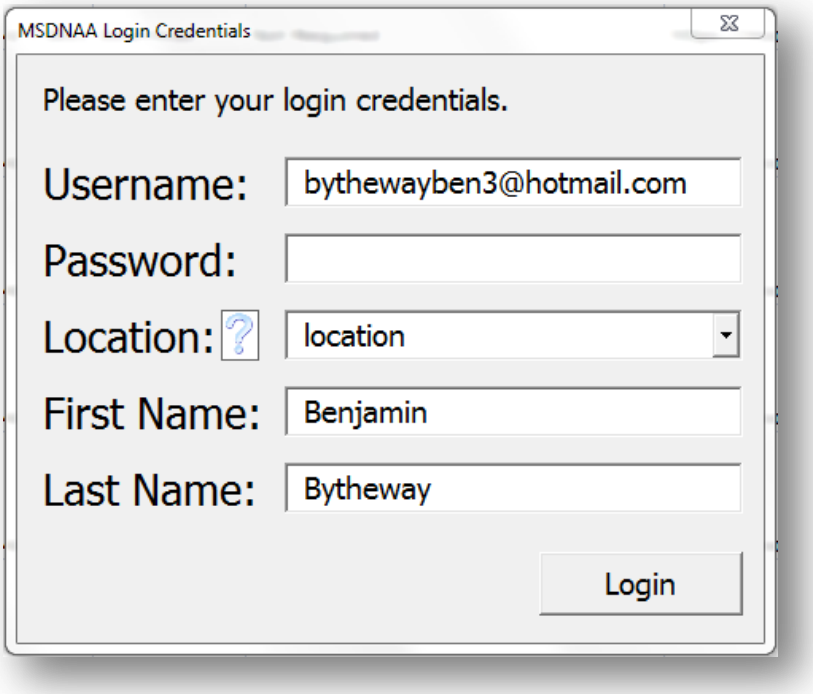
Logging in

The first thing to get around was the user log in. Because the site needs to be secure so that only authorized users can enter the site, it must be password protected.

When you first navigate to the site, you encounter the following login screen:



Normally the user would enter their username and password as credentials for the site. Since this would be difficult to manage, I found that the most user friendly solution to manage logging in was a user form:



Once the username and password are entered and the user clicks the “Login” button, the Excel application will:

1. Open up Internet Explorer
2. Browse to the login page
3. Insert the username and password into their respective fields and finally
4. Click the “Log me in!” button

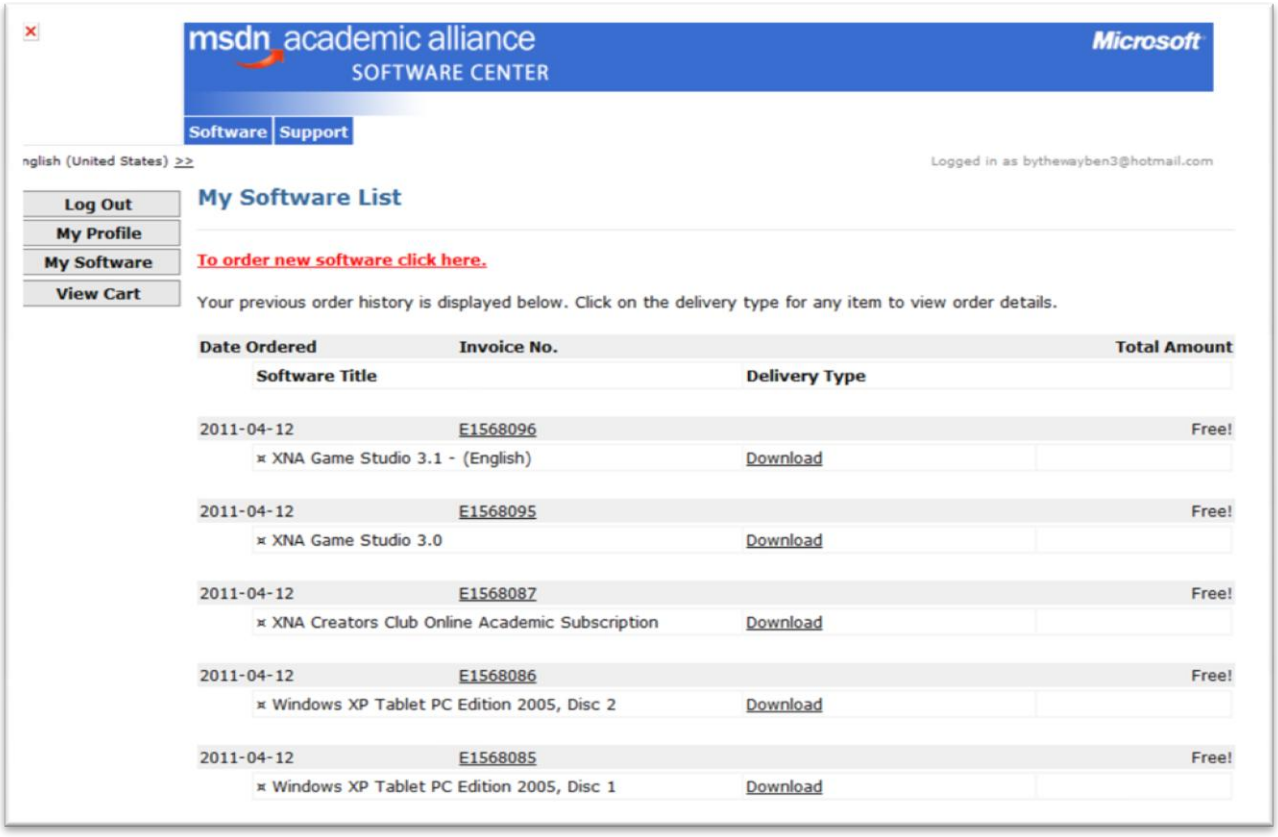
Once the button is clicked, the user is logged in and the application can access everything that it needs to.

Recording previously downloaded items

Once all of the authentication process is completed, the application will record any information about the products that the user previously purchased.

The reason why the application must do this is because there are users who may have already purchased multiple items from the site. Certainly the user would feel frustrated if the application could only record the future products that the user purchases. Therefore, we should first record all of these items before we start purchasing more.

The Excel application brings us to this screen where we see all of the users previously downloaded products.



The application will:

1. Gather all of the unique order id’s associated with each download

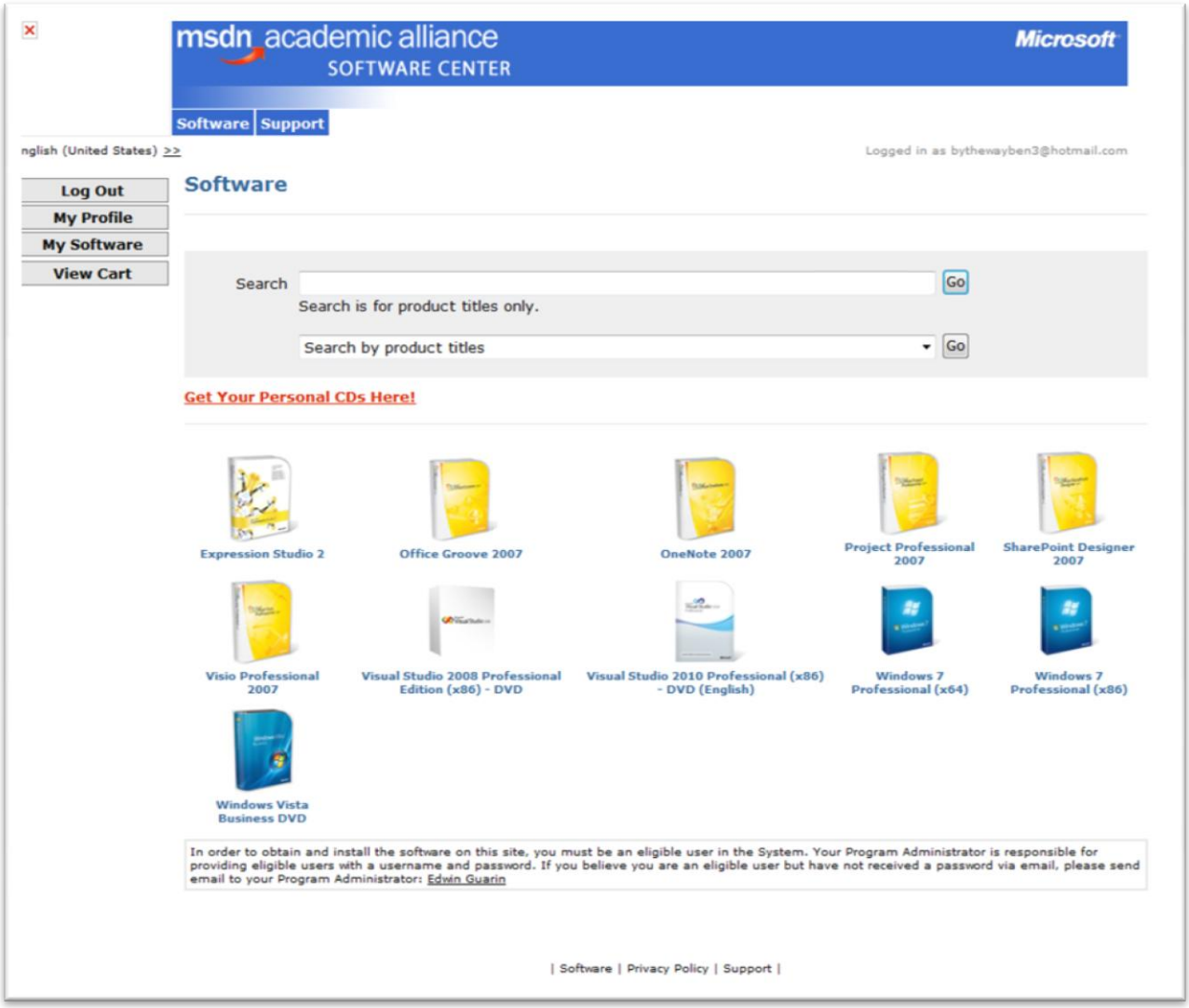
2. Loop through each of those ids and navigate to the download page of each (you may notice the “Download” link associated with each product.

As it loops through each of the products, it records the information about each of the products on the spreadsheet. (See section on Record Information for Product)

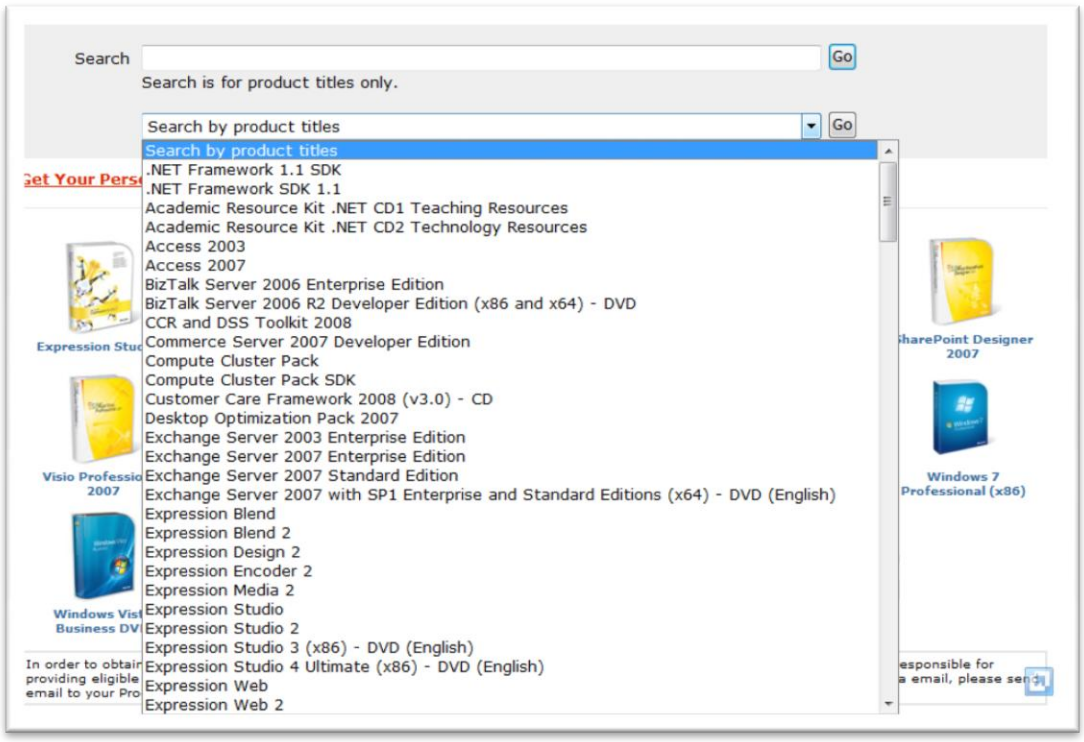
Adding products to online cart

Once we have recorded all of the information for the previously downloaded products our next step is to start adding products to the online cart.

The application will navigate to the home page to grab all of the products that are possible to download:

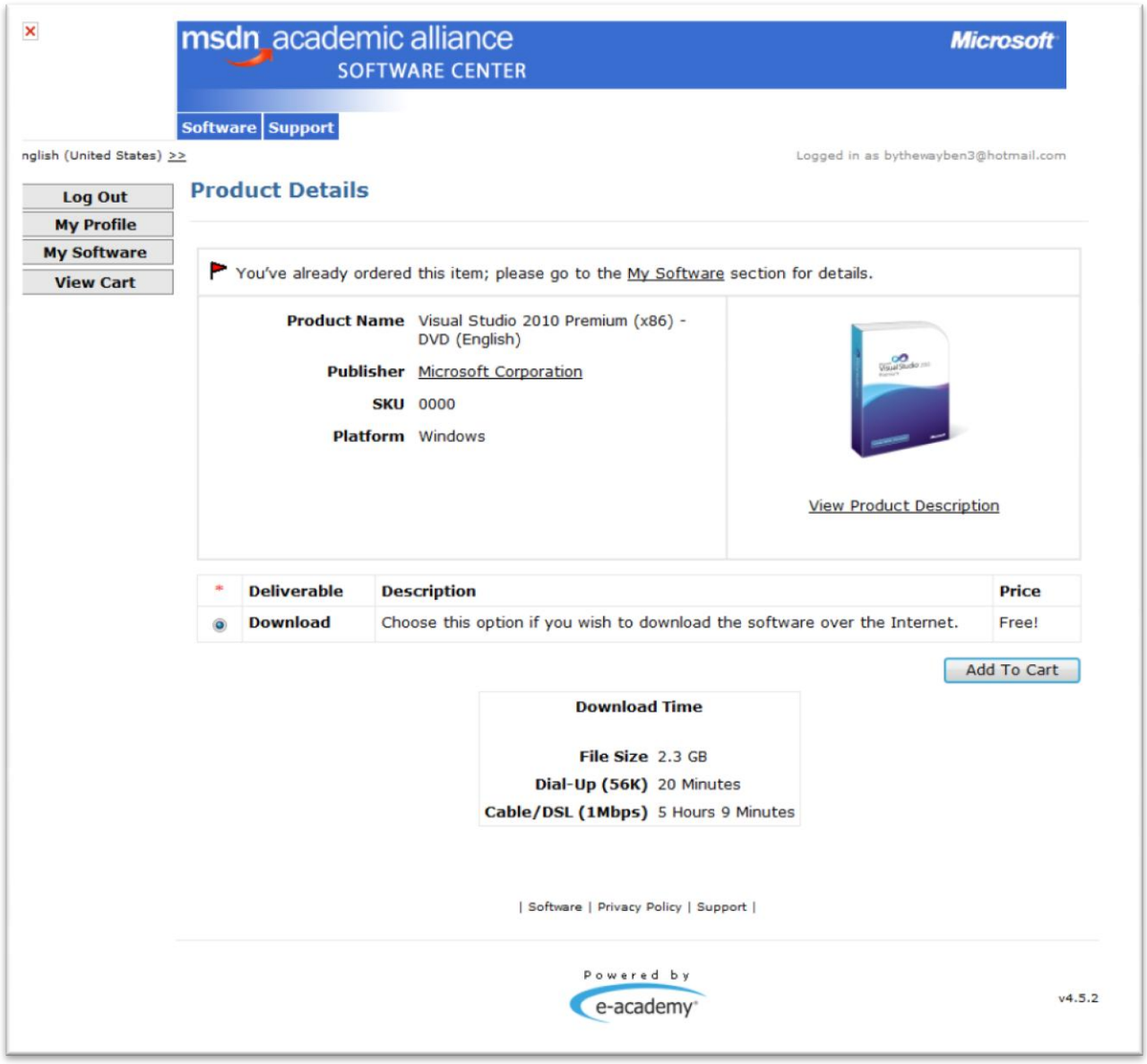


After the application navigates to the page, it will do some regular expressions to find the id’s of all of the products that are possible to download. It finds them by going through the drop-down box options and extracting the information from the values.



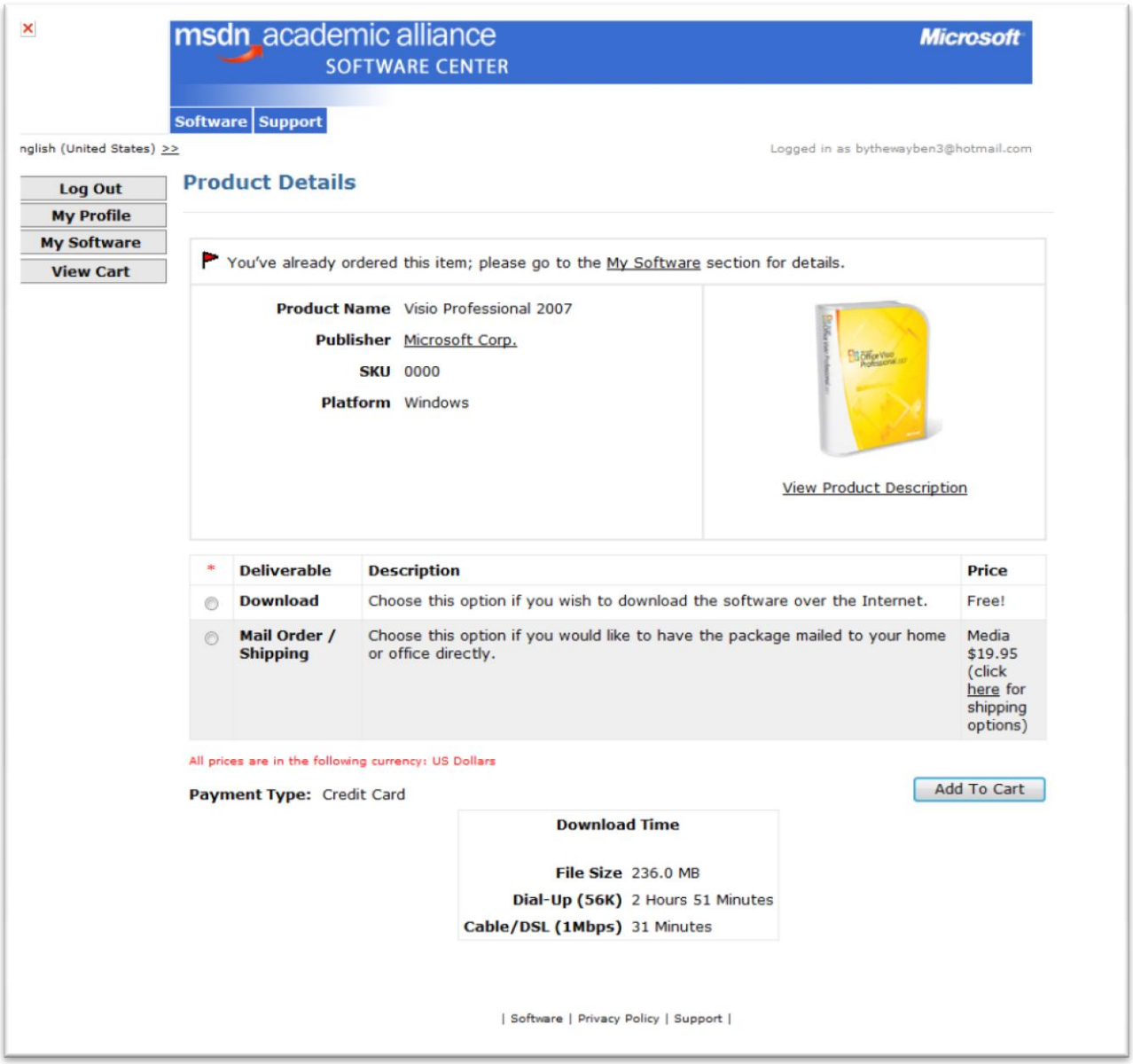
After it extracts each product id it will loop through each of the product id's and add each product to the cart.

It first navigates to that products details page:

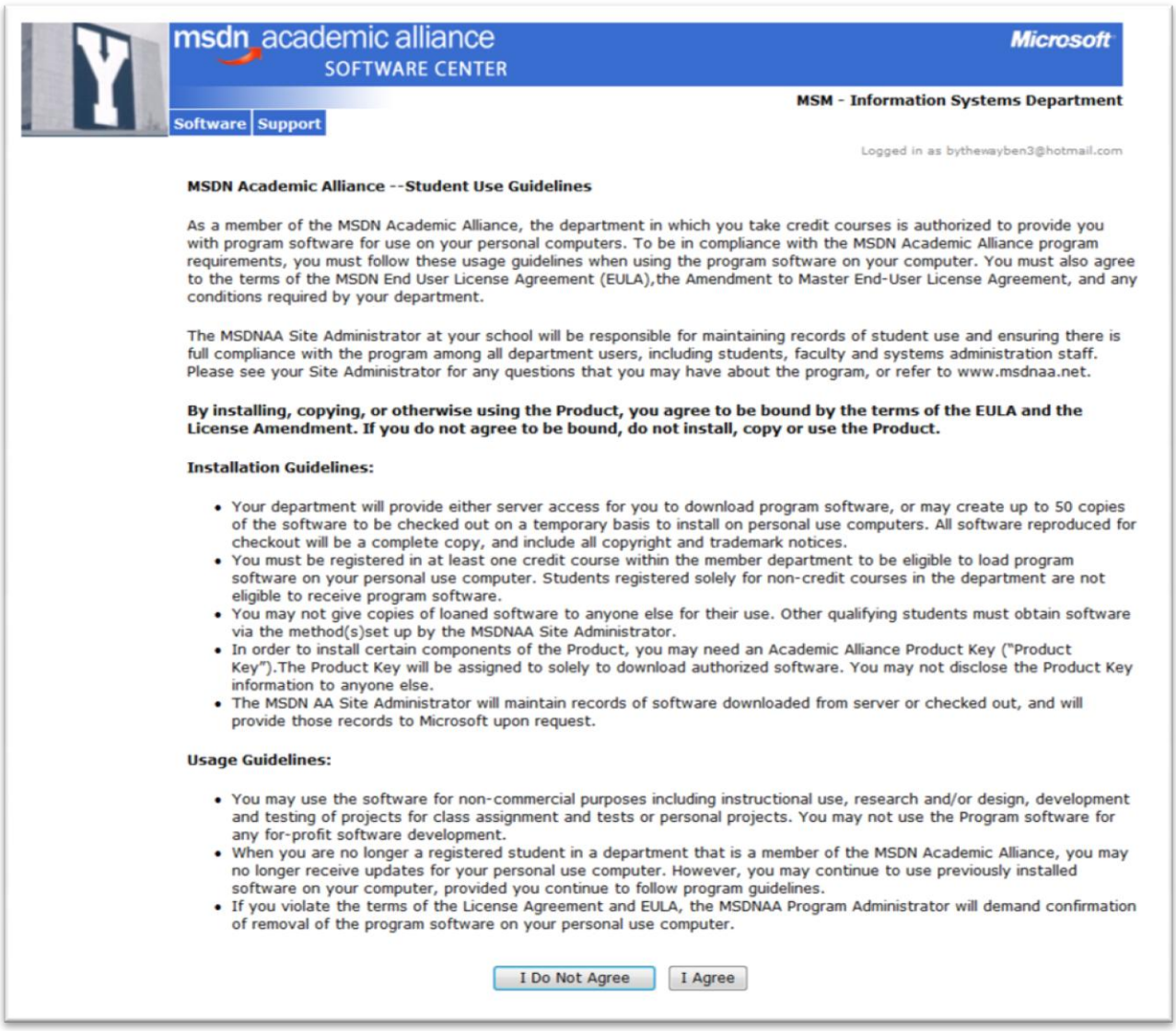


The application then checks to see if the product was already ordered (notice the red flag below "Product Details". If it was already ordered, it will simply go onto the next product. If the product wasn't ordered, the application goes through a few more steps to make sure that it will check out properly:

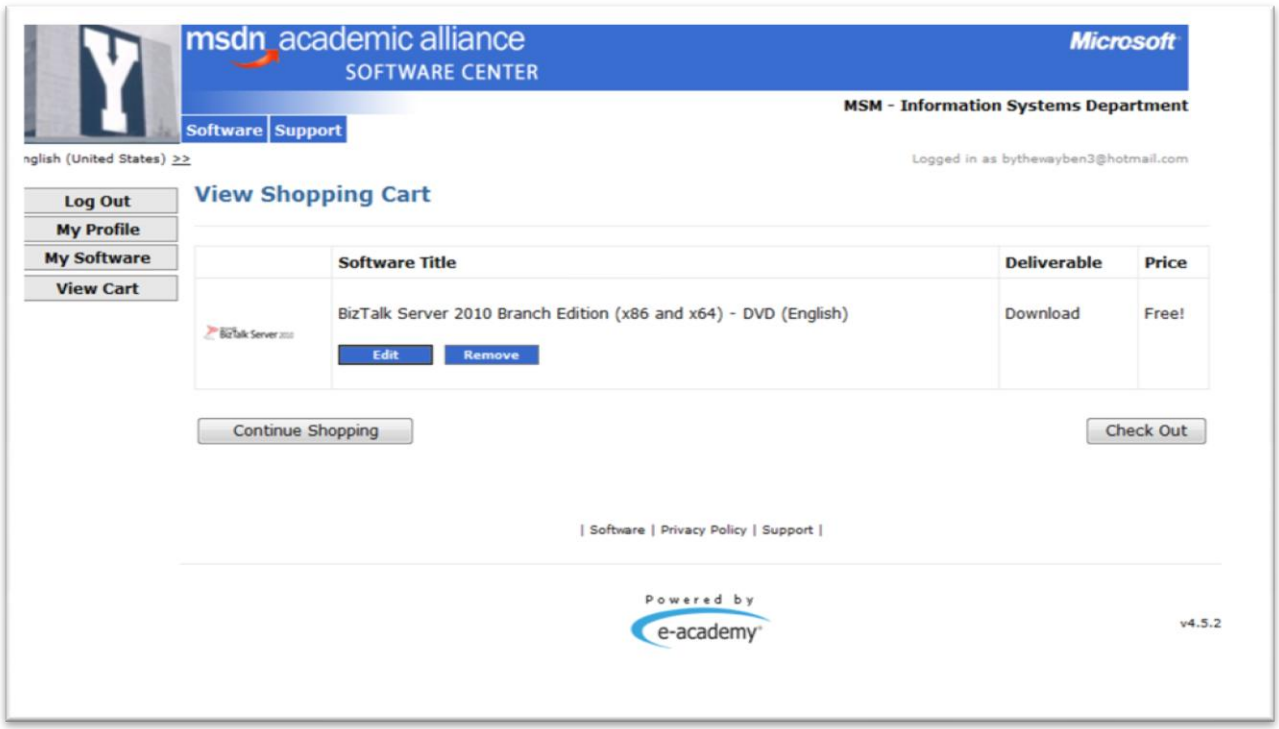
1. If there is an option to download or purchase a physical cd of the product, the application will check the "Download" option.



2. Once checked, it will click the "Add To Cart" button which will navigate to the End User License Agreement.



3. On the license agreement page, the application will click the "I Agree" button. One thing to note is that not every product has a license agreement page. Some just simply go to the checkout page. Either way the user will end up on the "View Shopping Cart" page.

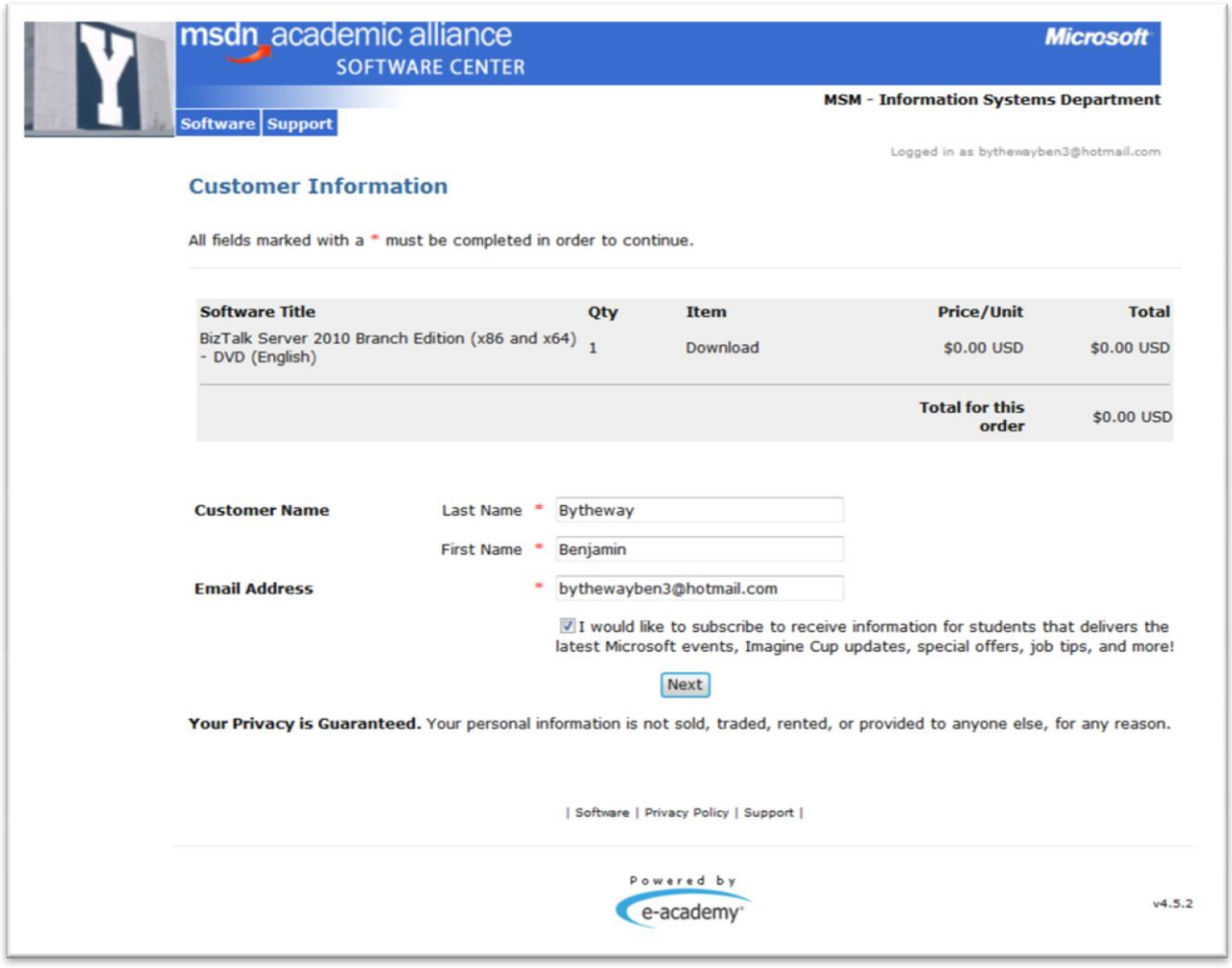


This finally concludes the process of adding a product to the cart.

Checking Out

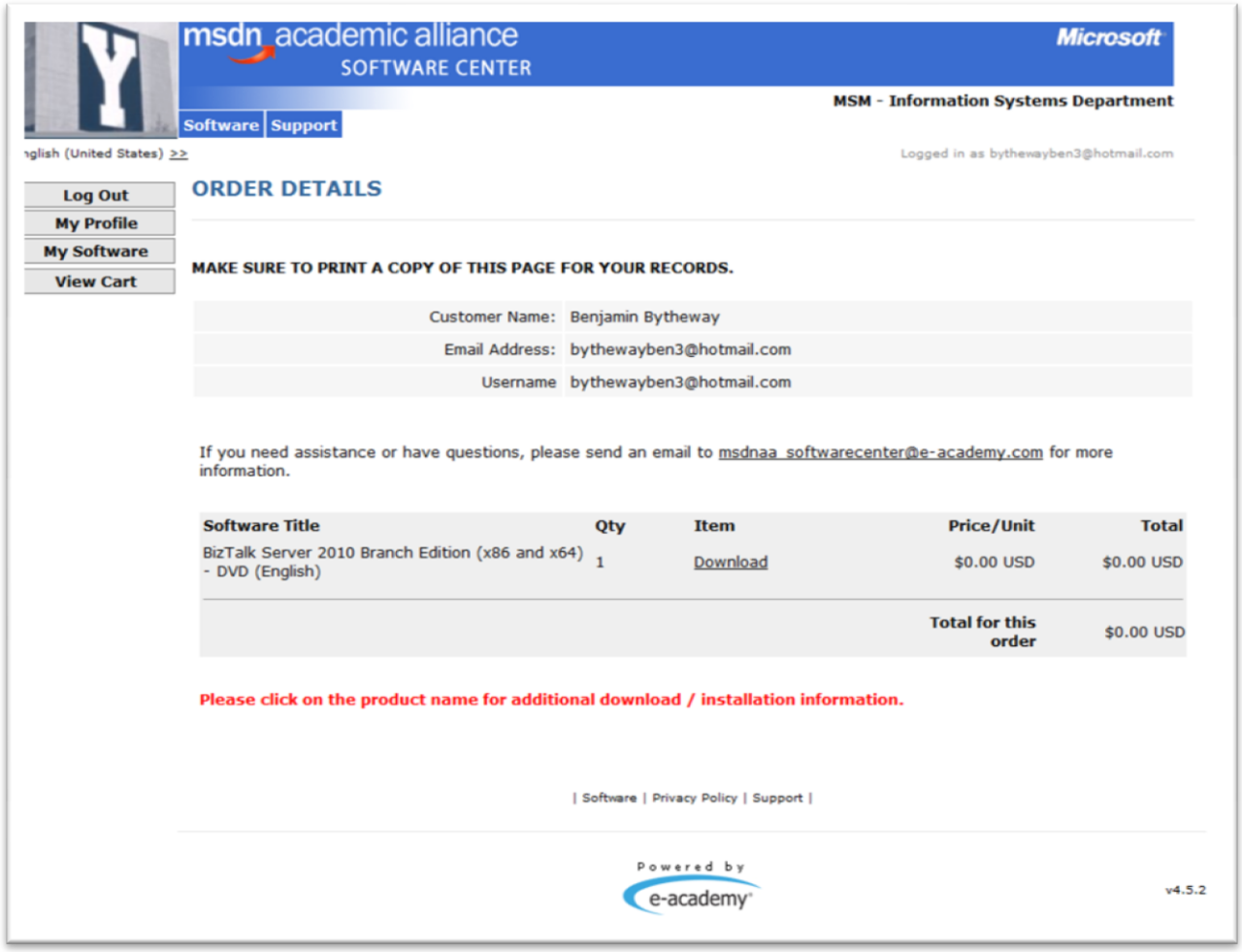
Once the product or products are in the shopping cart, we need to check out.

On the above “View Shopping Cart” screen you may notice the “Check Out” button. The application will click that button which will navigate to the order information page:



The application has already asked the user for their last name and their first name (refer back to the Login user form), so the application simply fills in the values for those two fields.

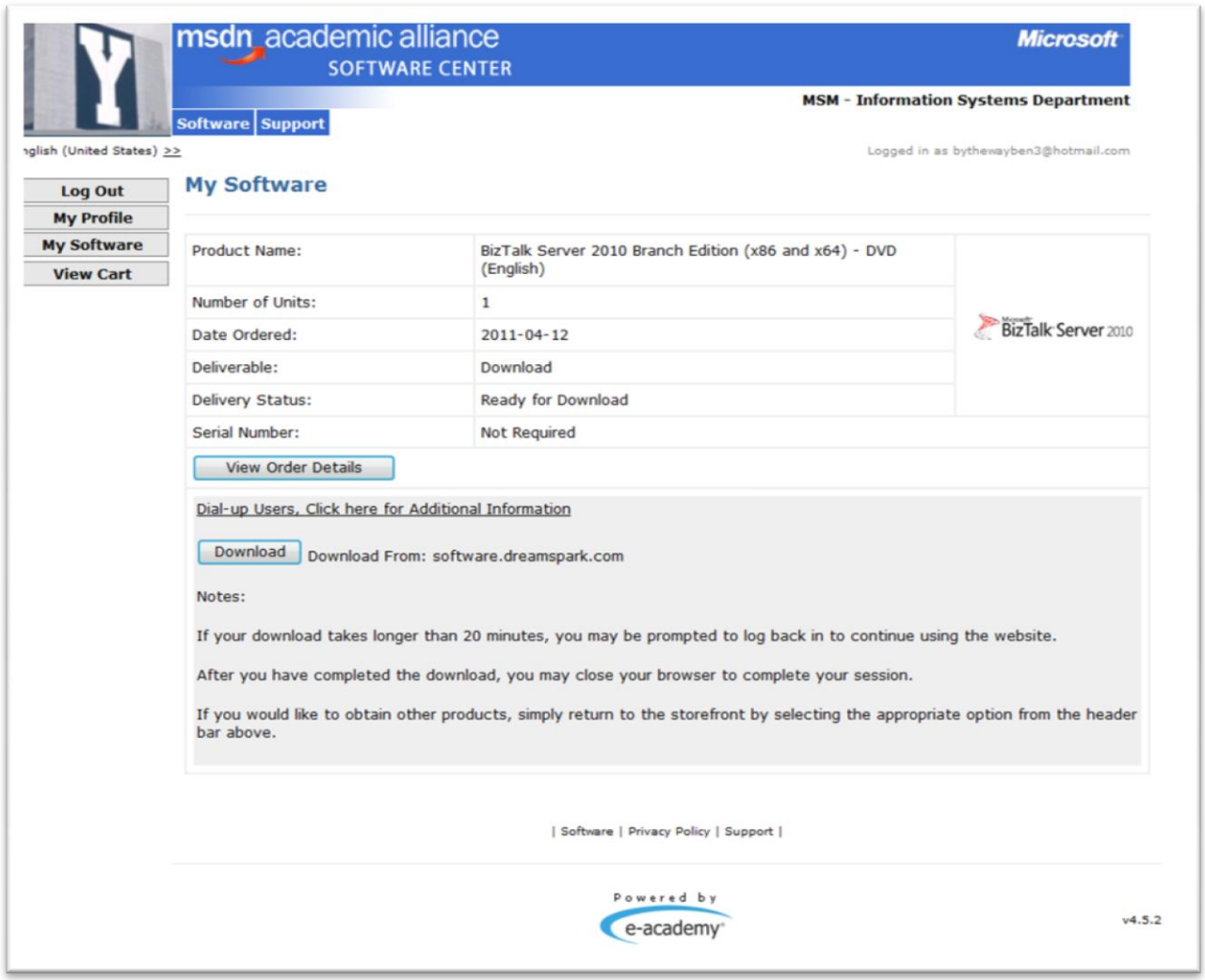
Once that information is filled in, it will click the “Next” button. This will bring up the “Order Details” page:



That concludes the process of ordering the item(s).

Record Information for Product

Once the product is ordered, the product information will be recorded. You may notice on the above screen that says “Download”. That link will take you to the product details page. The application will click that link which will lead the user to the product details page. If they are not already on that page, the application will navigate to that page:



Once on this page, the application will do the following:

1. Record information about the product
    a. Order Item ID
    b. Product Name
    c. Number of Units
    d. Date Ordered
    e. Deliverable type (which will always be “Download”)
    f. Serial Number
    g. Download URL
2. Download the image

Once all of those things are done the application will simply insert all of it into the Excel spreadsheet in row format.

All of this text information is recorded by either using regular expressions on the source, or by asking for the value of the text in the node in the DOM. All of the images are downloaded by using the IE agent that was given to us in class.

Discussion of learning and conceptual difficulties encountered

This section will include both the things that I learned through the project and also some of the difficulties that I encountered while programming the application.

What I Learned

There are multiple things that I learned while doing my project.

First, I learned that visual basic is a very outdated language and needs to be updated. There were so many times that I felt frustrated with the Visual Basic editor. It ended up working out in the end being that visual basic is a very useful language.

Second, I learned how to automate internet explorer by using the internet explorer agent we learned about in class. I learned how to use this tool to automate things that would otherwise be tedious and very annoying to use.

Third, I learned some regular expressions. There were multiple times that I needed to use regular expressions to finish a task. I was able to grow my knowledge and ability of regular expressions.

Fourth, I learned how to input images into an excel spreadsheet. That was something that I didn't know how to do before this class and even before this project. However, through this project I was able to learn how to save images and then use them in my excel spreadsheet.

Finally, I learned more than just how to use the internet explorer agent to automate web browsing and gathering information, but I learned how I could use JavaScript and regular HTML DOM objects to get information that I needed.

Difficulties I Encountered

There were a few snags I encountered while I did my project as well.

First, I ran into a problem when trying to do regular expressions. I didn't know what kind of library Excel uses for regular expressions. I also didn't know if it was built in, or if I had to use a COM object or something to actually create the regular expressions. It was actually pretty easy to find out. A quick search on Google provided me with the answer.

Second, I ran into a problem of knowing how to use the password characters. I actually played around with the text input box for a while and found the password char field. I figured that I would put in an asterisk and see what happened. That ended up doing exactly what I wanted it to do, so it wasn't really hard to figure out either.

One of the hard things to figure out was how to check for null values of an object in the DOM. One of the tasks of the application requires that I check for the existence of certain values. For example, I had to check to see if there was a certain button that existed on one of the pages. Sometimes it would appear and sometimes it wouldn't. But when I do a getElementById for the given id of the button, it throws a runtime exception. That ended up being a problem because I need to get some kind of value in the variable I was assigning it to, even if it was a null value. So, what happened is I had to do some searching online and I finally found a blog post that explained how that worked. It was a hard issue because the internet explorer automation isn't something that is "built in" to Excel. You must instantiate an excel object before you can take advantage of the feature. So, that was something that was a little bit hard to find, but I finally found a solution that would make the variable null if the given element id wasn't found on the page. That was really good news for me because that meant that I could check to see whether the given element existed on the page without crashing the application.

Finally the thing that was a difficult issue to figure out was in getting the images. There was at least one product on the MSDNAA website that had an invalid image URL. The image was either missing or it didn't ever exist in the first place. So, when I would go to download the image, it wasn't always guaranteed that I would get back an image. When I would try to save it, no errors would stop the execution of the code, but then when I would subsequently try to insert the image into the page, it would fail. The reason is because I never downloaded the image into the spreadsheet. The solution was to find out how to fail silently and continue execution of the code. Through this bug, I was able to learn the three ways of handling errors.